View metadata, citation and similar papers at core.ac.uk
brought to you by CORE

УДК 004.942(045)

Боровик В.М., канд. техн. наук

ПРОЕКТУВАННЯ МОДЕЛЕЙ РОЗПОДІЛЕНОЇ ОБРОБКИ ДАНИХ ІЗ ВІРТУАЛЬНИМ СЕРВЕРОМ

Інститут комп'ютерних технологій
Національного авіаційного університету

Досліджено модель серверів баз даних як системи масового обслуговування з пріоритетами. Визначається метод оптимізації пріоритетів та принципи практичного застосування результатів дослідження в системах "клієнт-сервер" з розподіленою обробкою даних

Вступ

Моделі "клієнт-сервер" дозволили будувати гнучкі системи розподіленої обробки даних. Дана концепція дозволяє розміщувати сервера та клієнтів як на окремих комп'ютерах, так і на одному, де система одночасно є сервером і клієнтом [1]. Все це породжує проблеми їх взаємодії як у технічному аспекті побудови комп'ютерних мереж, так і у прикладному – при побудові інформаційних систем та її важливої частини – системи баз даних.

Розглядаючи сучасні методи розрахунку та оптимізації параметрів традиційних чи мультисервісних систем та систем зв'язку, можна визначити аналогії у підході до моделі як системи масового обслуговування (СМО) з пріоритетами. Загальним підходом дослідження таких систем є марковські моделі прийняття рішення [2].

Мета статті

Предметом нашого дослідження є моделі серверів баз даних з віртуальним сервером ("*virtual server*"), де клієнти підключаються не до реального серверу, а до його проміжної складової – диспетчера [3]. Перевагою такої системи є гнучкість управління запитами при можливості оптимізації їх доступу до серверу. Дається метод формалізації управління такою системою, як СМО на основі теорії марковських процесів. Кінцевою метою є розробка алгоритму роботи диспетчера по управлінню запитами.

Постановка задачі

У період розробки перших систем управління базами даних (СУБД) технологія "клієнт-сервер" тільки зароджувалася, тому в архітектурі систем не було адекватного механізму організації взаємодії процесів типу "клієнт" і процесів типу "сервер". У сучасних СУБД цей механізм є фактично головним і від ефективності його реалізації залежить ефективність роботи системи в цілому. Рамки статті не дозволяють розглянути еволюцію типів організації подібних механізмів. В основному, цей механізм визначається структурою реалізації серверних процесів і відноситься до проблематики архітектури серверу баз даних. Поняття "серверний процес", яке ми використовуємо, ширше визначає технологію обробки даних як програмно-технічний комплекс.

Для обслуговування більшого числа клієнтів на сервері повинно бути запущено відповідне число одночасно працюючих серверних процесів, а це різко підвищує вимоги до ресурсів ЕОМ, на якій запускаються серверні процеси. Особливо це стосується моделей "один-до-одного", тобто коли один сервер обслуговує одного клієнта. Можливість роботи одного серверу з декількома клієнтами дозволяє в повній мірі використовувати розділяємі об'єкти (починаючи з відкритих файлів і закінчуючи даними з системних каталогів), що значно зменшує потреби в пам'яті та загальне число процесів операційної системи. Однак таке рішення має свої недоліки. Якщо сервер може виконуватися тільки на одному процесорі (*CPU*),

з'являється природне обмеження на застосування СУБД для мультипроцесорних платформ. Якщо комп'ютер має, наприклад, чотири процесори, то СУБД з одним сервером використовує тільки один з них, не задіючи решту. Така проблема вирішується уводом проміжного диспетчера ("*virtual server*") (рис. 1).

У вказаній архітектурі, яка є предметом дослідження даної статті, клієнти підключаються не до реального серверу, а до проміжного ланцюга, який називається диспетчером.

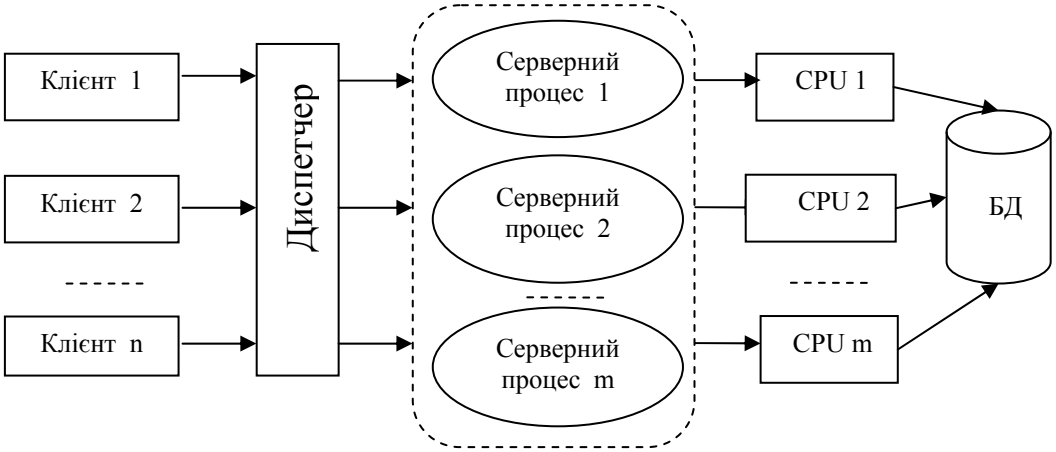

Рис. 1. Архітектура з віртуальним сервером

Математична модель

Розглянемо спочатку для більшої наглядності проміжну модель з диспетчером одно-серверної архітектури, де від клієнтів поступають заяви (запити) різного типу, кількість типів яких – n.

Усі заяви (запити) поступають на прилад (сервер чи серверний процес) у деякі випадкові моменти часу $t_0 < t_1 < t_2 \ldots < t_k \ldots < \ldots$. Інтервал між їх появою $\tau_k = t_k - t_{k-1}$ є незалежним випадковим значенням з експоненціальним законом розподілу

$$F(t) = P\{\tau_k \le t\} = 1 - \mathrm{e}^{-\lambda t},$$

де $\dfrac{1}{\lambda}$ – середній час між послідовними моментами появи двох однотипних запитів.

Далі ми будемо розрізняти типи заяв від відповідного клієнта, а саме, $\lambda_i(\overline{1,n})$ – інтенсивність появи заяв i-го типу (від i-го клієнта). Те, що кожний клієнт може посилати заяви тільки одного типу не є сильним обмеженням. Подібну модель ми отримуємо у випадку, коли один віртуальний клієнт посилає запити різних типів. У подальшому модель поступово буде ускладнюватися. Таким чином, на сервер поступають n пуассонівських вхідних потоків різнотипних запитів.

Усі заяви, які поступають на прилад, можуть бути виконані на будь-якому з них, при чому час їх виконання буде випадковим значенням з щільністю вірогідності $\varphi(t) = \mu \mathrm{e}^{-\mu t}$, де $\dfrac{1}{\mu}$ – середній час виконання залежить від типу заяви. У загальному випадку при багатоплатформеній архітектурі час виконання залежить від типу запиту та номера приладу (серверу, серверного процесу), тому $\mu_{ij}(i = \overline{1,n}; j = \overline{1,m})$. В нашому випадку при однотипних серверах час виконанн запиту не залежить від номеру прибору, тому $\mu_i(i = \overline{1,n})$.

Випадковий характер вказаних показників використовується при аналізі завантаження системи.

Для баз даних виконання процедури обробки запитів залежить від об'єму ін-

формації та складності алгоритму обробки, які задіяні у кожному випадку, тому одна і таж процедура може виконуватися різний час. Крім того, пуассонівський розподіл часу доступу і обробки запитів дає критичні по завантаженню параметри системи, що гарантує успішну роботу системи у реальному часі. Запити на обробку від клієнта до серверу поступають тільки у вигляді команд ініціалізації процедур, які разом з даними знаходяться на сервері.

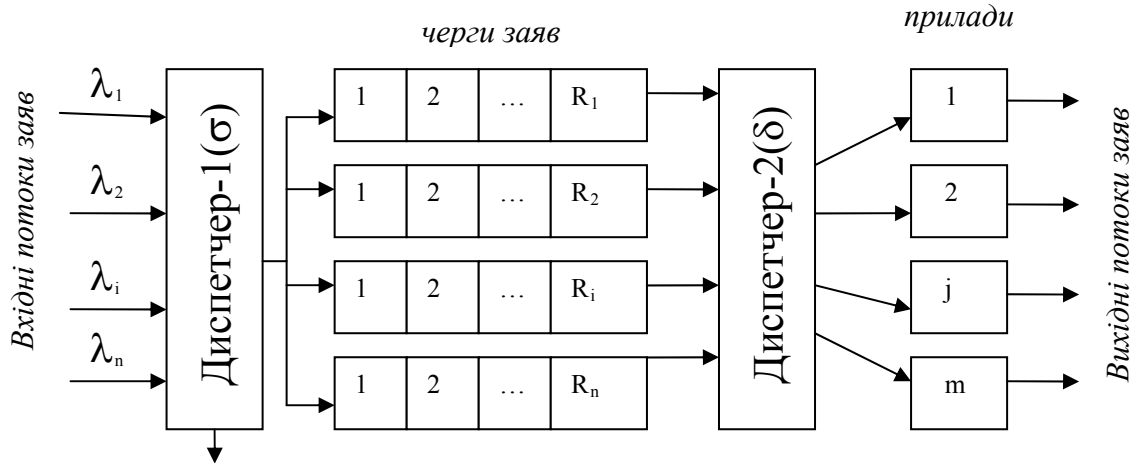

Рис. 2. Структура моделі СМО з віртуальним сервером

Якщо прилад вільний (може прийняти нашу заяву), то вона зразу приймається на виконання. Також просто вирішується питання, якщо в черзі є заяви одного типу. Розглядається ситуація управління заявами, які очікують своєї обробки. Фізична інтерпретація черги має інформаційний характер і може пояснюватися по-різному. У нашому випадку деталі цього механізму не мають принципового значення. Протоколи обробки заяв постійно ускладнюються, а загальна тенденція – пошук оптимальних методів контролю роботи системи з урахуванням усіх можливих критеріїв. Наша задача - дати приклад рішення вибору запиту для обробки на сервері при альтернативних

При n>1 треба вже вирішувати альтернативні ситуації призначення заяв на обслуговування (виконання) на відповідному приладі. Для цього, щоб формально визначити механізм вибору введемо поняття простору станів системи. Простір станів системи описується векторами $(\vec{k},\vec{i})$, де $\vec{k}=(k_1,k_2,...,k_m)$ – номера типів заяв, які обслуговуються на кожному з m приборів, $\vec{i}=(i_1,i_2,\ldots,i_m)$ – кількість заяв по кожному s типу в черзі. В подальшому індексований параметр "i" буде визначати кількість заяв даного типу в черзі, а без індексу – просто номер заяви.

При $\vec{k}\Big|_{k_j=0,(j=\overline{1,m})}$ прилад вільний,

при $\vec{i}\Big|_{i_s=0,(s=\overline{1,n})}$ – у черзі відсутні заяви s-го типу. Обмеження на довжину черги задаються двома параметрами:

1) для роздільної черги – по кожному типу заяв $i_s \le r_s,(s=\overline{1,n})$, де r_s – максимально можливий розмір черги для заяв s-го типу;

2) для загальної черги всіх типів заяв – $\sum_{s=1}^{n} i_s \le R$, де R – обмеження по розміру загальної черги.

При появі сигналу «прилад вільний» може статися ситуація, коли в черзі знаходяться заяви більше одного типу. Для вирішення таких конфліктних ситуацій використовується пріоритетний параметр $\delta_s^j(\vec{k}\Big|_{k_{sj}=0,}\vec{i}),(s=\overline{1,n};j=\overline{1,m})$, визнача-

ючий можливість вибору заяви *s*-го типу на *j*-й прилад. Це функції «Диспетчера-2» на рис. 2.

Досліджуємі моделі з віртуальним сервером можуть враховувати альтернативні ситуації, які з'являються при відсутності місць для очікування в черзі. Дійсно при заповненій черзі треба вирішити питання – чи поставити нову заяву замість однієї з черги іншого типу, чи дати їй відмову. Попадання у чергу є обов'язковою умовою обслуговування заяви, тому існує можливість її не обслужити на даному відрізку часу. Таким чином, включення найбільш цінних заяв у чергу є гарантією їх участі при прийнятті рішення про призначення чергової заяви на обслуговування до даної групи приладів, що суттєво при оптимізації функціонування усієї системи. Для цього вводиться пріоритетний параметр $\vartheta_s^t(\vec{k},\vec{i}),(s,t=\overline{1,n};s\neq t)$, визначаючий можливість виштовхування заяви *t*-го типу заявою *s*-типу. Це функції «Диспетчера-1».

Для досліджуваних зазначеного класу моделей розглядається робота одного диспетчера для постанови на прилад (сервер) без можливості виштовхування черги та обмеженням на загальну чергу чи окремі черги по кожному типу. В повному вигляді цей механізм може застосовується при введенні відповідних умов. Далі використовується апарат марковських процесів, який дозволяє у стаціонарних режимах роботи розробляти ясні у інтерпретації детерміновані алгоритми керування такими системами. Відомо, що випадковий процес є марковським, якщо для усіх фазових траєкторій ймовірність попадання системи у деякий наступний стан залежить тільки від того, в якому стані вона знаходиться в даний час і не залежить від попередньої історії.

Будемо розглядати систему у моменти розмноження при попаданні у чергу пуассонівських вхідних потоків , маючих властивість відсутності післядії.

На етапах "загибелі"(сидіння чи перескоків), тобто етапів їх обслуговування, час якого розподілений по експоненціальному закону, одержуємо ланцюг, маючий марковську властивість.

Для однорідного марковського ланцюгу, тобто того, де відсутня залежність стану від часу, одержуємо для неприводимого випадку (кожний стан може бути досягнений із будь-якого іншого) можливість досягнення межі $\lim_{t\to\infty} P_{ik}(t)=\pi_k$, не залежного від початкового стану ланцюга, де $P_{ik}(t)$ – ймовірність попадання системи у стан "*k*", якщо до того вона перебувала у стані "*i*". Множина $\{\pi_k\},(k=\overline{1,K})$ є граничний розподіл ймовірностей станів, називаємих далі стаціонарними ймовірностями станів. Ланцюг у такому вигляді є ергодичним.

Для визначення стаціонарних ймовірностей станів використовується графсхема можливих переходів марковського процесу, при цьому кругами відображаються всі можливі стани процесу, а стрільками – можливі перехідні інтенсивності (умовні та безумовні). Після цього для збалансованого випадку складаються рівняння для усіх стаціонарних ймовірностей станів.

У нашій моделі сервер інтерпретується як множина однотипних приладів, кількість яких визначається можливістю серверу одночасно обробляти *m* заяв для виконання процедури (запиту) – тип заяви, час виконання процедури – час виконання заяви. В принципі можливо інтерпретувати таку систему й як мультипроцесорну.

Стаціонарна ймовірність стану $(\vec{k},\vec{i})$ задається через $\pi(\vec{k},\vec{i})$, де вектор $\vec{i}=(i_1,i_2,\ldots,i_m)$ задає число заяв в черзі по кожному типу та $\vec{k}=(k_1,k_2,...,k_m)$ – номера типів заяв, які обслуговуються на відповідному приладі.

При неальтернативних ситуаціях на етапах "розмноження" поступають пуассонівські вхідні потоки заяв, при тому виконуються обмеження на число місць у черзі. У момент закінчення обслуговування неальтернативна ситуація відпові-

дає існуванню у черзі заяв тільки одного типу. Для цієї ситуації, яка відноситься до тривіального випадку моделей "розмноження та загибелі", стан системи являє собою лінійний ланцюг. Закон вибору заяв на обслуговування у цьому випадку може бути вільним, що не змінює розподіл стаціонарних ймовірностей станів системи. Для визначеності будемо задавати, що для черги з заявами одного типу буде виконуватися дисципліна обслуговування *FCFS* – "перший прийшов – перший обслугований".

При надходженні інформації "прилад вільний" може з'явитися ситуація, коли у черзі знаходяться заяви більше одного типу. Для вирішення таких конфліктних ситуацій використовується пріоритетний параметр: $\delta^s(\vec{I}),(s=\overline{1,n})$, який визначає можливість вибору заяви s- го типу для обслуговування на приладі. Звичайно, що якась заява обов'язково буде вибрана, тому :

$$\sum_{s=1}^{n} \delta^{s}(i_1, i_2, ..., i_n) u(i_s) = 1,$$

де

$$u(x) = \begin{cases} 1, \text{ ÿêùî} \quad x > 0, \\ 0, \text{ ÿêùî} \quad x \le 0. \end{cases}.$$

Тепер ми можемо у векторній формі записати систему рівнянь для моделі $M_n|M_1|r_i(i=\overline{1,n})$ чи R:

$$\pi(0,\vec{0})\sum_{s=1}^{n}\lambda_s = \sum_{s=1}^{n}\mu_s\pi(s,\vec{0}); \qquad (1)$$

$$\pi(k,\vec{0})[\sum_{s=1}^{n}\lambda_s+\mu_k]=\pi(0,\vec{0})\lambda_k+\sum_{s=1}^{n}\mu_s\pi(s,\vec{0}\big|_{0_k=1}); \quad (2)$$

$$\pi(k,\vec{i})[\sum_{s=1}^{n}\lambda_s u(\Omega^s)+\mu_k]=$$

$$\sum_{s=1}^{n}u(i_s)\lambda_s\pi(k,\vec{i}\big|_{i_s=i_s+1})+$$

$$u(\Omega^k)[u(\sum_{\substack{t=1\\t\neq k}}^{n}i_t)\delta^k(\vec{i}\big|_{i_k=i_k+1})+$$

$$+[1-u(\sum_{\substack{t=1\\t\neq k}}^{n}i_t)]]\sum_{s=1}^{n}\mu_s\pi(s,\vec{i}\big|_{i_k=i_k+1}); \quad (3)$$

$$k=\overline{1,n};\ \sum_{s=1}^{n}i_s>0;\ i_s=\overline{0,\Omega^s};\ s=\overline{1,n};$$

$$\Omega^s=\begin{cases}\dfrac{(r_s-i_s),(s=\overline{1,m})\text{ - ðîçä³ëüí³ ÷åðãè}}{R-\sum_{s=1}^{n}i_s\ \text{ - çàãàëüí³ ÷åðãè}}.\end{cases} \quad (4)$$

Нормуюча умова буде записана у такому вигляді:

$$\pi(0,\vec{0})+\sum_{k=1}^{n}\sum_{\vec{i}}\pi(k,\vec{i})=1. \qquad (5)$$

В одержаній системі рівнянь $u(\Omega^s)$ задає обмеження на число місць у черзі.

Запис $\pi(k,\vec{i}\big|_{i_s=i_{s-1}}$ визначає перехід із стану $(k,i_1,i_2,\ldots,i_s,\ldots,i_n)$ у $(k,i_1,i_2,\ldots, i_{s-1},\ldots,i_n)$, а $\pi(k,\vec{i}\big|_{i_s=i_s+1})$ із стану $(k,i_1,i_2, \ldots,i_s,\ldots,i_n)$ у $(k,i_1,i_2,\ldots,i_{s+1},\ldots,i_n)$, що означає вихід заяви із черги чи її вхід.

Критерій ефективності

Досліджуючи механізм зміни стану системи, були показані можливі переходи із одного стану у інший без урахування оцінки ефективності їх функціонування по тому чи іншому алгоритму. Для визначення оптимальності функціонування треба ввести критерії ефективності.

Одержані у результаті рішення задачі значення $\delta(\cdot)$ і будуть шукаємими змінними параметрами управління на відміну від фіксованих $\vec{\lambda},\vec{\mu},R,r_i(i=\overline{1,n})$.

У якості критерію виберемо мінімізацію часу перебування заяв у черзі (втрата від чекання). Для визначення цього критерію введемо показник витрат із-за перебування l заявок s-го типу у черзі –

β_l^s. Тоді сумарні втрати у одиницю часу у системі від чекання будуть складати :

$$L = \sum_{s=1}^{n} \sum_{l=1}^{R} \beta_l^s \sum_{\vec{k}} \sum_{\vec{i}} \pi(\vec{k}, \vec{i}\,)\Big|_{i_s = l}. \qquad (6)$$

Із опису рівнянь ми бачимо , що у критеріях ефективності існує тільки лінійна залежність від $\pi(\cdot)$, а нелінійна складові (множина $\pi(\cdot)$ на $\delta(\cdot)$) входять в обмеження для усіх стаціонарних ймовірностей стану системи. Для вирішення цієї проблеми використовується стандартний прийом заміни існуючих нелінійних складових, що дозволить у подальшому використати методи лінійного програмування. Введення вказаних додаткових змінних потребує й нових обмежень з урахуванням конкретної моделі.

Визначення оптимальних пріоритетів

Головним математичним об'єктом досліджень даної роботи є СМО з кінцевим числом можливих станів у стаціонарному режимі. Процес керування такими системами розглядається, як пошук керуючих параметрів беззупинного марковського ланцюга.

Рівняння для стаціонарних ймовірностей моделі – (1)-(5). Критерій ефективності у загальному вигляді – (6).

Із теорії кінцевих ланцюгів Маркова відомо, що усі стаціонарні ймовірності станів $\pi(\cdot)>0$, тобто більше нуля. Оптимальне значення керуючих елементів – $\delta(\cdot)$ можуть бути рівні тільки 0 чи 1. Таким чином , оптимізаційна задача зводиться до пошуку екстремуму функції:

$$\begin{aligned} &L^* = f(\pi^*(\cdot)) = \max\{L = f(\pi(\cdot)) : \\ &Q(\pi(\cdot), \delta(\cdot)) = 0; \\ &0 < \pi(\cdot) < 1; \\ &\delta(\cdot) \in \{0,1\}; \sum_{P} \pi(\cdot) = 1\}, \end{aligned} \qquad (7)$$

де $\pi^*(\cdot)$ – шукаємі оптимальні стаціонарні ймовірності стану системи; Q – характеризує функціональний зв'язок стаціонарних ймовірностей стану та керуємих пріоритетів $\delta(\cdot)$; L^* – оптимальне значення критерію ефективності; P – множина усіх можливих стаціонарних ймовірностей стану, визначаємих типом та розмірністю моделі.

Головним привабливим результатом теорії марковських процесів є оптимальне управління їм як детермінірованим процесом, що дозволяє вирішувати задачу як відому задачу лінійного програмування. Це дозволяє, на перший погляд, складну математичну задачу звести до простої практичної реалізації.

Висновки

Розглянута методологія може бути використана для побудови пріоритетного управління запитами у системах клієнт-сервер. Результати розрахунків представляються у вигляді налаштованої таблиці переходів чи бази даних ситуаційних пріоритетів, які попередньо розраховуються по параметрам системи. При більш потужних системах із сервером додатків ця задача вирішується у реальному часі.

Такий підхід дозволяє розглядати систему як багаторівневу [4]. Подібну концепцію обробки даних пропагандують фірми *Oracle, Sun, Borland* та ін.

Список літератури

1. *Дейт Дж.* Введение в системы баз данных, 8-е изд. – М.: Вильямс, 2006. – 1328 с.

2. *Меликов А.З., Пономаренко Л.А., Паладюк В.В.* Телетрафик: Модели, методы,оптимизация. – К.: Политехника, 2007. – 256 с.

3. *Карпова Т.* Базы данных :модели, разработка, реализация. – СПб.: Питер, 2003. – 304 с.

4. *Петров В.Н.* Информационные системы. – СПб.: Питер, 2002. – 688 с.